Database Schema - PostgreSQL 15

Version: 1.0
Date: 2025-11-26
Status: Implementation Ready
Database: PostgreSQL 15+

---

1. Overview

Este esquema está diseñado para:

- ✅ **Compliance bancario**: PCI-DSS, GDPR (pseudonimización)
- ✅ **Performance**: UUIDv7 (sortable), índices optimizados, JSONB para flexibilidad
- ✅ **Auditoría**: Audit log separado con rotación agresiva
- ✅ **Event Sourcing**: Outbox pattern para garantía de eventos
- ✅ **Seguridad**: TDE encryption-at-rest, sin PII plain

---

2. Entity-Relationship Diagram

```
┌─────────────────────────────────────────────────────────────────┐
│                        signature_request                         │
├─────────────────────────────────────────────────────────────────┤
│ id (UUIDv7) PK                                                   │
│ customer_id (VARCHAR) NOT NULL -- pseudonymized                 │
│ transaction_context (JSONB) NOT NULL -- immutable business data │
│ status (VARCHAR) NOT NULL -- PENDING, CHALLENGE_SENT, etc.      │
│ active_challenge_id (UUID) FK NULLABLE                          │
│ created_at (TIMESTAMPTZ) NOT NULL                               │
│ updated_at (TIMESTAMPTZ) NOT NULL                               │
│ expires_at (TIMESTAMPTZ) NOT NULL                               │
└─────────────────┬───────────────────────────────────────────────┘
                  │ 1
                  │
                  │ N
┌─────────────────┴───────────────────────────────────────────────┐
│                     signature_challenge                          │
├─────────────────────────────────────────────────────────────────┤
│ id (UUIDv7) PK                                                   │
│ signature_request_id (UUID) FK NOT NULL                         │
│ channel_type (VARCHAR) NOT NULL -- SMS, PUSH, VOICE, BIOMETRIC │
│ provider (VARCHAR) NOT NULL -- TWILIO, PUSH_SERVICE, etc.       │
│ provider_challenge_id (VARCHAR)                                 │
│ provider_proof (TEXT) -- cryptographic receipt (non-repudiation)│
│ status (VARCHAR) NOT NULL -- PENDING, SENT, COMPLETED, FAILED   │
│ sent_at (TIMESTAMPTZ)                                           │
│ responded_at (TIMESTAMPTZ)                                      │
│ expires_at (TIMESTAMPTZ) NOT NULL                               │
│ error_code (VARCHAR)                                            │
│ raw_response (TEXT)                                             │
│ created_at (TIMESTAMPTZ) NOT NULL                               │
└─────────────────────────────────────────────────────────────────┘

┌─────────────────────────────────────────────────────────────────┐
│                         routing_rule                             │
├─────────────────────────────────────────────────────────────────┤
│ id (UUIDv7) PK                                                   │
│ name (VARCHAR) NOT NULL UNIQUE                                  │
│ description (TEXT)                                              │
│ priority (INTEGER) NOT NULL -- lower = higher priority          │
│ condition (TEXT) NOT NULL -- SpEL expression                    │
│ target_channel (VARCHAR) NOT NULL                               │
│ enabled (BOOLEAN) NOT NULL DEFAULT true                         │
│ created_at (TIMESTAMPTZ) NOT NULL                               │
│ updated_at (TIMESTAMPTZ) NOT NULL                               │
│ created_by (VARCHAR) -- admin user                              │
│ updated_by (VARCHAR)                                            │
└─────────────────────────────────────────────────────────────────┘

┌─────────────────────────────────────────────────────────────────┐
│                      connector_config                            │
├─────────────────────────────────────────────────────────────────┤
│ id (UUIDv7) PK                                                   │
│ provider (VARCHAR) NOT NULL UNIQUE                              │
│ enabled (BOOLEAN) NOT NULL DEFAULT true                         │
│ config (JSONB) NOT NULL -- timeout, retry, etc.                 │
│ vault_path (VARCHAR) NOT NULL -- path to secrets in Vault       │
│ degraded_mode (BOOLEAN) DEFAULT false                           │
│ degraded_since (TIMESTAMPTZ)                                    │
│ error_rate (NUMERIC(5,2)) -- for circuit breaker                │
│ last_health_check (TIMESTAMPTZ)                                 │
│ created_at (TIMESTAMPTZ) NOT NULL                               │
│ updated_at (TIMESTAMPTZ) NOT NULL                               │
└─────────────────────────────────────────────────────────────────┘

┌─────────────────────────────────────────────────────────────────┐
│                         outbox_event                             │
├─────────────────────────────────────────────────────────────────┤
│ id (UUIDv7) PK                                                   │
│ aggregate_id (UUID) NOT NULL -- signature_request_id            │
│ aggregate_type (VARCHAR) NOT NULL -- 'SignatureRequest'     │
│ event_type (VARCHAR) NOT NULL -- SIGNATURE_REQUEST_CREATED, etc.│
│ payload (JSONB) NOT NULL -- event data (no PII)               │
│ payload_hash (VARCHAR) -- SHA-256 for integrity               │
│ created_at (TIMESTAMPTZ) NOT NULL                             │
│ published_at (TIMESTAMPTZ) -- set by Debezium after publish   │
└─────────────────────────────────────────────────────────────┘


┌─────────────────────────────────────────────────────────────┐
│                         audit_log                             │
│                (separate tablespace + rotation)               │
├─────────────────────────────────────────────────────────────┤
│ id (UUIDv7) PK                                                │
│ entity_type (VARCHAR) NOT NULL                                │
│ entity_id (UUID) NOT NULL                                     │
│ action (VARCHAR) NOT NULL -- CREATE, UPDATE, DELETE           │
│ actor (VARCHAR) NOT NULL -- user or system                    │
│ changes (JSONB) -- before/after snapshot (no PII)             │
│ ip_address (VARCHAR)                                          │
│ user_agent (VARCHAR)                                          │
│ created_at (TIMESTAMPTZ) NOT NULL                             │
└─────────────────────────────────────────────────────────────┘
```

---

3. DDL - Table Definitions

3.1 signature_request

```sql
CREATE TABLE signature_request (
    id UUID PRIMARY KEY,
    customer_id VARCHAR(255) NOT NULL,
    transaction_context JSONB NOT NULL,
    status VARCHAR(50) NOT NULL,
    active_challenge_id UUID,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMPTZ NOT NULL,

    CONSTRAINT chk_status CHECK (status IN (
        'PENDING', 'CHALLENGE_SENT', 'SIGNED', 'FAILED', 'EXPIRED', 'ABORTED'
    ))
);

-- Índices
CREATE INDEX idx_signature_request_customer_id ON signature_request(customer_id);
CREATE INDEX idx_signature_request_status ON signature_request(status);
CREATE INDEX idx_signature_request_created_at ON signature_request(created_at
DESC);
CREATE INDEX idx_signature_request_expires_at ON signature_request(expires_at)
    WHERE status NOT IN ('SIGNED', 'FAILED', 'EXPIRED');

-- GIN index para búsquedas en JSONB
CREATE INDEX idx_signature_request_context_gin ON signature_request
    USING GIN (transaction_context);

-- Comentarios para documentación
COMMENT ON TABLE signature_request IS 'Agregado raíz para solicitudes de firma
digital';
COMMENT ON COLUMN signature_request.customer_id IS 'ID pseudonimizado del cliente
(NO PII)';
COMMENT ON COLUMN signature_request.transaction_context IS 'Contexto inmutable de
transacción en JSONB';
COMMENT ON COLUMN signature_request.active_challenge_id IS 'Único challenge activo
a la vez';
```

3.2 signature_challenge

```
CREATE TABLE signature_challenge (
    id UUID PRIMARY KEY,
    signature_request_id UUID NOT NULL,
    channel_type VARCHAR(50) NOT NULL,
    provider VARCHAR(100) NOT NULL,
    provider_challenge_id VARCHAR(255),
    provider_proof TEXT,
    status VARCHAR(50) NOT NULL,
    sent_at TIMESTAMPTZ,
    responded_at TIMESTAMPTZ,
    expires_at TIMESTAMPTZ NOT NULL,
    error_code VARCHAR(100),
    raw_response TEXT,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT fk_challenge_signature_request
        FOREIGN KEY (signature_request_id)
        REFERENCES signature_request(id)
        ON DELETE CASCADE,

    CONSTRAINT chk_channel_type CHECK (channel_type IN (
        'SMS', 'PUSH', 'VOICE', 'BIOMETRIC'
    )),

    CONSTRAINT chk_challenge_status CHECK (status IN (
        'PENDING', 'SENT', 'COMPLETED', 'FAILED', 'EXPIRED'
    ))
);

-- Índices
CREATE INDEX idx_challenge_signature_request ON
signature_challenge(signature_request_id);
CREATE INDEX idx_challenge_status ON signature_challenge(status);
CREATE INDEX idx_challenge_provider ON signature_challenge(provider);
CREATE INDEX idx_challenge_created_at ON signature_challenge(created_at DESC);

-- Index for finding active challenge
CREATE UNIQUE INDEX idx_active_challenge_per_request
    ON signature_challenge(signature_request_id)
    WHERE status IN ('PENDING', 'SENT');

COMMENT ON TABLE signature_challenge IS 'Desafíos enviados por diferentes
canales';
COMMENT ON COLUMN signature_challenge.provider_proof IS 'Recibo criptográfico del
provider (no-repudio legal)';
COMMENT ON COLUMN signature_challenge.raw_response IS 'Respuesta cruda del
provider para debugging';
```

3.3 routing_rule

```
CREATE TABLE routing_rule (
    id UUID PRIMARY KEY,
    name VARCHAR(255) NOT NULL UNIQUE,
    description TEXT,
    priority INTEGER NOT NULL,
    condition TEXT NOT NULL,
    target_channel VARCHAR(50) NOT NULL,
    enabled BOOLEAN NOT NULL DEFAULT true,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    created_by VARCHAR(255),
    updated_by VARCHAR(255),

    CONSTRAINT chk_priority_positive CHECK (priority >= 0),
    CONSTRAINT chk_target_channel CHECK (target_channel IN (
        'SMS', 'PUSH', 'VOICE', 'BIOMETRIC'
    ))
);

-- Índice para búsqueda ordenada por prioridad (short-circuit evaluation)
CREATE INDEX idx_routing_rule_priority
    ON routing_rule(priority ASC)
    WHERE enabled = true;
```

```sql
-- Índice único para evitar reglas duplicadas con mismo nombre
CREATE UNIQUE INDEX idx_routing_rule_name_unique
    ON routing_rule(LOWER(name));

COMMENT ON TABLE routing_rule IS 'Reglas de enrutamiento con expresiones SpEL';
COMMENT ON COLUMN routing_rule.priority IS 'Menor valor = mayor prioridad (first
match wins)';
COMMENT ON COLUMN routing_rule.condition IS 'Expresión SpEL evaluada contra
TransactionContext';
```

3.4 connector_config

```sql
CREATE TABLE connector_config (
    id UUID PRIMARY KEY,
    provider VARCHAR(100) NOT NULL UNIQUE,
    enabled BOOLEAN NOT NULL DEFAULT true,
    config JSONB NOT NULL,
    vault_path VARCHAR(500) NOT NULL,
    degraded_mode BOOLEAN DEFAULT false,
    degraded_since TIMESTAMPTZ,
    error_rate NUMERIC(5, 2) DEFAULT 0.00,
    last_health_check TIMESTAMPTZ,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT chk_error_rate CHECK (error_rate >= 0 AND error_rate <= 100),
    CONSTRAINT chk_degraded_consistency CHECK (
        (degraded_mode = false AND degraded_since IS NULL) OR
        (degraded_mode = true AND degraded_since IS NOT NULL)
    )
);

-- Índice para búsqueda de proveedores habilitados
CREATE INDEX idx_connector_enabled ON connector_config(enabled) WHERE enabled =
true;

COMMENT ON TABLE connector_config IS 'Configuración de conectores a providers
externos';
COMMENT ON COLUMN connector_config.vault_path IS 'Path a credenciales en HashiCorp
Vault';
COMMENT ON COLUMN connector_config.degraded_mode IS 'Provider pausado
temporalmente (error_rate >50%)';
COMMENT ON COLUMN connector_config.error_rate IS 'Tasa de error para Circuit
Breaker';
```

3.5 outbox_event

```
CREATE TABLE outbox_event (
    id UUID PRIMARY KEY,
    aggregate_id UUID NOT NULL,
    aggregate_type VARCHAR(100) NOT NULL,
    event_type VARCHAR(100) NOT NULL,
    payload JSONB NOT NULL,
    payload_hash VARCHAR(64),
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    published_at TIMESTAMPTZ
);

-- Índice para Debezium CDC
CREATE INDEX idx_outbox_created_at ON outbox_event(created_at ASC)
    WHERE published_at IS NULL;

-- Índice para búsqueda por agregado
CREATE INDEX idx_outbox_aggregate ON outbox_event(aggregate_id, aggregate_type);

-- Índice por tipo de evento
CREATE INDEX idx_outbox_event_type ON outbox_event(event_type);

COMMENT ON TABLE outbox_event IS 'Outbox pattern para eventos con garantía de
entrega';
COMMENT ON COLUMN outbox_event.payload_hash IS 'SHA-256 del TransactionContext
(sin PII)';
COMMENT ON COLUMN outbox_event.published_at IS 'Timestamp cuando Debezium publicó
a Kafka';
```

3.6 audit_log

```
-- Crear tablespace separado para audit logs (facilita rotación)
-- CREATE TABLESPACE audit_tablespace LOCATION '/var/lib/postgresql/audit';

CREATE TABLE audit_log (
    id UUID PRIMARY KEY,
    entity_type VARCHAR(100) NOT NULL,
    entity_id UUID NOT NULL,
    action VARCHAR(50) NOT NULL,
    actor VARCHAR(255) NOT NULL,
    changes JSONB,
    ip_address VARCHAR(45),
    user_agent VARCHAR(500),
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT chk_action CHECK (action IN ('CREATE', 'UPDATE', 'DELETE',
'ACCESS'))
) -- TABLESPACE audit_tablespace
PARTITION BY RANGE (created_at);

-- Particiones mensuales para rotación eficiente
CREATE TABLE audit_log_2025_11 PARTITION OF audit_log
    FOR VALUES FROM ('2025-11-01') TO ('2025-12-01');

CREATE TABLE audit_log_2025_12 PARTITION OF audit_log
    FOR VALUES FROM ('2025-12-01') TO ('2026-01-01');

-- Índices en particiones
CREATE INDEX idx_audit_log_entity ON audit_log(entity_type, entity_id);
CREATE INDEX idx_audit_log_actor ON audit_log(actor);
CREATE INDEX idx_audit_log_created_at ON audit_log(created_at DESC);

COMMENT ON TABLE audit_log IS 'Audit trail completo con rotación mensual';
COMMENT ON COLUMN audit_log.changes IS 'Snapshot before/after sin PII';
```

---

4. Data Types and Constraints

4.1 UUIDv7 Generation

```
-- Función para generar UUIDv7 (sortable by time)
-- UUIDv7 = timestamp (48 bits) + version (4 bits) + random (74 bits)

CREATE OR REPLACE FUNCTION uuid_generate_v7()
RETURNS UUID AS $$
DECLARE
    unix_ts_ms BIGINT;
    uuid_bytes BYTEA;
BEGIN
    unix_ts_ms := FLOOR(EXTRACT(EPOCH FROM clock_timestamp()) * 1000);

    uuid_bytes :=
        substring(int8send(unix_ts_ms) FROM 3 FOR 6) ||
        gen_random_bytes(10);

    uuid_bytes := set_byte(uuid_bytes, 6, (get_byte(uuid_bytes, 6) & 15) | 112);
    uuid_bytes := set_byte(uuid_bytes, 8, (get_byte(uuid_bytes, 8) & 63) | 128);

    RETURN encode(uuid_bytes, 'hex')::UUID;
END;
$$ LANGUAGE plpgsql VOLATILE;
```

```
COMMENT ON FUNCTION uuid_generate_v7() IS 'Genera UUIDv7 sortable con timestamp
embebido';
```

Uso en aplicación (Java):

```
// Usar librería: com.github.f4b6a3:uuid-creator:5.3.2
UUID id = UuidCreator.getTimeOrderedEpoch(); // UUIDv7
```

4.2 JSONB Schema Validation (PostgreSQL 15)

```
-- Validación de schema para transaction_context
CREATE DOMAIN transaction_context_jsonb AS JSONB
CHECK (
    VALUE ? 'customerId' AND
    VALUE ? 'amount' AND
    VALUE ? 'currency' AND
    VALUE ? 'transactionType' AND
    (VALUE->>'amount')::NUMERIC > 0
);

-- Aplicar a la tabla (opcional, puede ralentizar inserts)
-- ALTER TABLE signature_request
--     ALTER COLUMN transaction_context TYPE transaction_context_jsonb;
```

4.3 TTL Enforcement (Challenge Expiration)

```
-- Job periódico para limpiar challenges expirados (ejecutar cada 5 min)
CREATE OR REPLACE FUNCTION expire_old_challenges()
RETURNS INTEGER AS $$
DECLARE
    expired_count INTEGER;
BEGIN
    WITH updated AS (
        UPDATE signature_challenge
        SET status = 'EXPIRED'
        WHERE status IN ('PENDING', 'SENT')
          AND expires_at < CURRENT_TIMESTAMP
        RETURNING id
    )
    SELECT COUNT(*) INTO expired_count FROM updated;

    RETURN expired_count;
END;
$$ LANGUAGE plpgsql;

-- Crear extensión pg_cron para scheduling
-- CREATE EXTENSION pg_cron;
-- SELECT cron.schedule('expire-challenges', '*/5 * * * *',
--     'SELECT expire_old_challenges();');
```

5. Indexes Strategy

5.1 Performance Indexes

```
-- Hot path: buscar signature_request por ID (PK automático)
-- Hot path: buscar challenges por signature_request_id
CREATE INDEX CONCURRENTLY idx_challenge_request_id
    ON signature_challenge(signature_request_id);

-- Query común: buscar requests por customer
CREATE INDEX CONCURRENTLY idx_request_customer_status
    ON signature_request(customer_id, status);

-- Query común: buscar requests recientes
CREATE INDEX CONCURRENTLY idx_request_recent
    ON signature_request(created_at DESC)
    WHERE status NOT IN ('EXPIRED', 'FAILED');

-- Analytics: count por provider
CREATE INDEX CONCURRENTLY idx_challenge_provider_status
    ON signature_challenge(provider, status);
```

5.2 GIN Indexes para JSONB

```
-- Búsqueda por campos dentro del transaction_context
CREATE INDEX CONCURRENTLY idx_request_context_gin
    ON signature_request USING GIN (transaction_context jsonb_path_ops);

-- Ejemplo de query que usa este índice:
-- SELECT * FROM signature_request
-- WHERE transaction_context @> '{"riskLevel": "HIGH"}';
```

5.3 Partial Indexes (Performance Optimization)

```
-- Índice solo para requests activos (ahorra espacio y mejora performance)
CREATE INDEX CONCURRENTLY idx_request_active
    ON signature_request(id, status)
    WHERE status IN ('PENDING', 'CHALLENGE_SENT');

-- Índice solo para challenges fallidos (debugging)
CREATE INDEX CONCURRENTLY idx_challenge_failed
    ON signature_challenge(provider, error_code, created_at DESC)
    WHERE status = 'FAILED';
```

---

6. Security Constraints

6.1 Pseudonymization Enforcement

```
-- Trigger para prevenir inserción de PII (ejemplo: detectar emails en
customer_id)
CREATE OR REPLACE FUNCTION check_no_pii()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.customer_id ~* '[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}' THEN
        RAISE EXCEPTION 'PII detected in customer_id: email not allowed';
    END IF;

    IF LENGTH(NEW.customer_id) < 16 THEN
        RAISE EXCEPTION 'customer_id must be pseudonymized (min 16 chars)';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_no_pii
    BEFORE INSERT OR UPDATE ON signature_request
    FOR EACH ROW
    EXECUTE FUNCTION check_no_pii();
```

6.2 Audit Trigger

```
CREATE OR REPLACE FUNCTION audit_changes()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO audit_log (
        id,
        entity_type,
        entity_id,
        action,
        actor,
        changes,
        created_at
    ) VALUES (
        uuid_generate_v7(),
        TG_TABLE_NAME,
        NEW.id,
        TG_OP,
        current_user,
        jsonb_build_object(
            'before', row_to_json(OLD),
            'after', row_to_json(NEW)
        ),
        CURRENT_TIMESTAMP
    );

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Aplicar a tablas críticas
CREATE TRIGGER trg_audit_routing_rule
    AFTER INSERT OR UPDATE OR DELETE ON routing_rule
    FOR EACH ROW
    EXECUTE FUNCTION audit_changes();

CREATE TRIGGER trg_audit_connector_config
    AFTER INSERT OR UPDATE OR DELETE ON connector_config
    FOR EACH ROW
    EXECUTE FUNCTION audit_changes();
```

7. Sample Data (Development)

```
-- Sample routing rules
INSERT INTO routing_rule (id, name, description, priority, condition,
target_channel, enabled, created_at, updated_at)
VALUES
    (uuid_generate_v7(),
     'High Risk Transactions',
     'Transacciones de alto riesgo siempre por SMS',
     10,
     'context.riskLevel == ''HIGH''',
     'SMS',
     true,
     CURRENT_TIMESTAMP,
     CURRENT_TIMESTAMP),

    (uuid_generate_v7(),
     'Large Amounts',
     'Montos mayores a 10K USD por Voice',
     20,
     'context.amount.value > 10000 && context.amount.currency == ''USD''',
     'VOICE',
     true,
     CURRENT_TIMESTAMP,
     CURRENT_TIMESTAMP),

    (uuid_generate_v7(),
     'Default Push',
     'Por defecto usar Push (más barato)',
     100,
     'true',
     'PUSH',
     true,
     CURRENT_TIMESTAMP,
     CURRENT_TIMESTAMP);

-- Sample connector config
INSERT INTO connector_config (id, provider, enabled, config, vault_path,
created_at, updated_at)
VALUES
    (uuid_generate_v7(),
     'TWILIO',
     true,
     '{"timeout_ms": 5000, "retry_attempts": 3, "from_number":
"+1234567890"}'::jsonb,
     'secret/signature-router/twilio',
     CURRENT_TIMESTAMP,
     CURRENT_TIMESTAMP),

    (uuid_generate_v7(),
     'PUSH_SERVICE',
     true,
     '{"timeout_ms": 3000, "retry_attempts": 2}'::jsonb,
     'secret/signature-router/push',
     CURRENT_TIMESTAMP,
     CURRENT_TIMESTAMP);
```

8. Database Maintenance

8.1 Vacuum Strategy

```
-- Configurar autovacuum agresivo para tablas de alto volumen
ALTER TABLE signature_request SET (
    autovacuum_vacuum_scale_factor = 0.05,
    autovacuum_analyze_scale_factor = 0.02
);

ALTER TABLE outbox_event SET (
    autovacuum_vacuum_scale_factor = 0.01,
    autovacuum_analyze_scale_factor = 0.01
);
```

8.2 Partition Management (Audit Log)

```
-- Script para crear particiones futuras (ejecutar mensualmente)
CREATE OR REPLACE FUNCTION create_next_audit_partition()
RETURNS VOID AS $$
DECLARE
    partition_date DATE;
    partition_name TEXT;
    start_date DATE;
    end_date DATE;
BEGIN
    partition_date := DATE_TRUNC('month', CURRENT_DATE + INTERVAL '1 month');
    partition_name := 'audit_log_' || TO_CHAR(partition_date, 'YYYY_MM');
    start_date := partition_date;
    end_date := partition_date + INTERVAL '1 month';

    EXECUTE format(
        'CREATE TABLE IF NOT EXISTS %I PARTITION OF audit_log FOR VALUES FROM (%L)
TO (%L)',
        partition_name,
        start_date,
        end_date
    );

    RAISE NOTICE 'Created partition: %', partition_name;
END;
$$ LANGUAGE plpgsql;
```

8.3 Data Retention

```sql
-- Eliminar datos antiguos (ejecutar semanalmente)
CREATE OR REPLACE FUNCTION purge_old_data()
RETURNS TABLE(table_name TEXT, deleted_count BIGINT) AS $$
BEGIN
    -- Eliminar signature_requests completados hace más de 90 días
    DELETE FROM signature_request
    WHERE status IN ('SIGNED', 'FAILED', 'EXPIRED')
      AND updated_at < CURRENT_DATE - INTERVAL '90 days';

    GET DIAGNOSTICS deleted_count = ROW_COUNT;
    table_name := 'signature_request';
    RETURN NEXT;

    -- Eliminar outbox_events publicados hace más de 30 días
    DELETE FROM outbox_event
    WHERE published_at IS NOT NULL
      AND published_at < CURRENT_DATE - INTERVAL '30 days';

    GET DIAGNOSTICS deleted_count = ROW_COUNT;
    table_name := 'outbox_event';
    RETURN NEXT;

    -- Eliminar particiones de audit_log antiguas (>365 días)
    -- Se deja como ejemplo manual para evitar borrado accidental
END;
$$ LANGUAGE plpgsql;
```

---

9. Performance Tuning

9.1 Connection Pool (HikariCP)

```yaml
spring:
  datasource:
    hikari:
      maximum-pool-size: 20
      minimum-idle: 5
      connection-timeout: 2000  # 2s según especificación
      idle-timeout: 600000      # 10 min
      max-lifetime: 1800000     # 30 min
      leak-detection-threshold: 60000  # 1 min
```

9.2 PostgreSQL Configuration

```
# postgresql.conf optimizations for banking workload

# Memory
shared_buffers = 4GB
effective_cache_size = 12GB
work_mem = 32MB
maintenance_work_mem = 512MB

# Checkpoints
checkpoint_timeout = 15min
checkpoint_completion_target = 0.9
max_wal_size = 4GB

# Query Planner
random_page_cost = 1.1  # SSD
effective_io_concurrency = 200

# Connections
max_connections = 200

# Logging (para auditoría bancaria)
logging_collector = on
log_statement = 'mod'  # Log todas las modificaciones
log_duration = on
log_min_duration_statement = 1000  # Log queries >1s
```

---

10. Encryption at Rest (TDE)

```
-- Habilitar TDE usando extensión pgcrypto (básico)
CREATE EXTENSION IF NOT EXISTS pgcrypto;

-- Opción avanzada: usar PostgreSQL Enterprise con TDE nativo
-- O usar LUKS encryption a nivel de filesystem para /var/lib/postgresql

-- Ejemplo de columna específica encriptada (si se requiere)
-- ALTER TABLE signature_challenge
--     ADD COLUMN provider_api_key_encrypted BYTEA;
```

Status: ✅ **COMPLETE - READY FOR FLYWAY MIGRATION**

Next Steps:

- Convertir a Flyway migrations (`V1__initial_schema.sql`, etc.)

- Configurar TDE según proveedor cloud (AWS RDS Encryption, etc.)
- Implementar JPA entities correspondientes